Knauf, Rainer; Gonzalez, Avelino J.; Abel, Thomas :

A framework for validation of rule-based systems

Zuerst erschienen in:
IEEE Transactions on Systems, Man and Cybernetics, Part B : Cybernetics, 32 (2002), Nr. 3, S. 281–295
DOI: 10.1109/TSMCB.2002.999805

A Framework for Validation of Rule-Based Systems

Rainer Knauf, Avelino J. Gonzalez, and Thomas Abel

***Abstract*—This paper describes a complete methodology for the validation of rule-based expert systems. This methodology is presented as a five-step process that has two central themes: 1) to create a minimal set of test inputs that adequately cover the domain represented in the knowledge base and 2) a Turing Test-like methodology that evaluates the system's responses to the test inputs and compares them to the responses of human experts.**

The development of minimal set of test inputs takes into consideration various criteria, both user-defined, and domain-specific. These criteria are used to reduce the potentially very large set of test inputs to one that is practical, keeping in mind the nature and purpose of the developed system.

The Turing Test-like evaluation methodology makes use of only one panel of experts to both evaluate each set of test cases and compare the results with those of the expert system, as well as with those of the other experts. The hypothesis being presented here is that much can be learned about the experts themselves by having them anonymously evaluate each other's responses to the same test inputs. Thus, we are better able to determine the validity of an expert system.

Depending on its purpose, we introduce various ways to express validity as well as a technique to use the validity assessment for the refinement of the rule base.

Lastly, the paper describes a partial implementation of the test input minimalization process on a small but nontrivial expert system. The effectiveness of the technique was evaluated by seeding errors into the expert system, generating the appropiate set of test inputs and determining whether the errors could be detected by the suggested methodology.

***Index Terms*—Expert system validation, rule-based systems, test case validation.**

I. Introduction

THERE is abundant evidence of the need for an integrated approach toward validation and verification of complex systems (cf. [8]) ranging from mathematically well-based formal approaches ([4]) and approaches that also use formal approaches but that are more driven by practical considerations [27] to high-level philosophical and psychological approaches focusing on human factor issues [23]. Newer trends in research are focused on technologies to accompany the system development by an integrated validation and verification (V&V) concept of all aspects at the different stages of development and implementation, starting at the high-level design and going down to the operational details [16] and on adapting technologies of database integrity checking to V&V of rule bases [9]. In [25], there is an overview on various directions of actual research in V&V of knowledge-based systems.

Boehm [5] as well as O'Keefe and O'Leary [22] developed a very intuitive approach that characterizes *verification* and *validation* as *building the system right* and *building the right system*, respectively. Both are considered a part of a general evaluation strategy (see [17] or [21]). This perspective is adapted here. Verification is basically the test of whether or not a system follows its (formal) specification. The present paper is focused on the validation issue and provides a methodology to get evidence that a given system really does what it should do in the eyes of experts and users. This methodology is constructed for a frequently used kind of rule-based systems. The rules are based HORN-Logic with single propositional expressions as their *then*-part and conjunctions of expressions in their *if*-part, which can be either single propositional expressions or comparsion expressions with attributes and values.

In [26], the authors clearly point out that "the inability to adequately evaluate systems may become the limiting factor in our ability to employ systems that our technology and knowledge will allow us to design."

There has been one quite comprehensive approach to the validation of knowledge-based systems described in the literature. This is the ESPRIT-II project VALID during 1989–1992, as surveyed in [20]. This project's goal was to undertake a comprehensive approach to the problem of Validation for existing knowledge-based systems (KBS). In order to do so, several methods for different validation issues were created and different expert systems were considered. The project's main result was a validation environment in which different expert systems can be validated. To relate this to our present endeavour, we need to make more explicit the validation concept that underlies the mentioned project.

Two points are important to relate our present work to the project mentioned. First, VALID makes enormously strict assumptions about the object to be validated. The high degree of assumed formal knowledge is nicely illustrated in [18], where a Petri net approach is invoked for validation within VALID. Second, after the completion of VALID, it has been recognized that there is still a flaw in testing methodologies: "It seems that testing is a mandatory step in the KBS validation. However, no substantiated testing methodology for KBS is available and often knowledge engineers are guideless in the testing phase" [20]. This paper is a contibution to bridge this gap.

The core objective of validation and verification (V&V) of an intelligent system is actually very simple: to ensure that, when provided with a legal set of inputs, the system will produce an

Manuscript received December 13, 2000; revised July 20, 2001, November 16, 2001, and December 27, 2001.

R. Knauf is with the Faculty of Computer Science and Automation, Technical University of Ilmenau, 98684 Ilmenau, Germany (e-mail: rainer.knauf@tu-ilmenau.de).

A. J. Gonzalez is with the Intelligent Systems Laboratory, School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816-2450 USA (e-mail: gonzalez@ucf.edu).

T. Abel is with the GFT Systems GmbH, 98694 Ilmenau, Germany (e-mail: thomas.abel@gft-systems.de).

Publisher Item Identifier S 1083-4419(02)03005-4.

answer, solution, or behavior that is equivalent to that provided by the best human experts.

For a large knowledge-based system, providing this assurance can be a daunting task that may require a significant effort on the part of the development team. However, its importance is such that it clearly deserves such a serious treatment, as an invalid expert system can at best lead to loss of credibility by the users and, at worst, to disastrous results.

We concentrate on the validation portion of the V&V problem, as that is the one more closely related to ensuring appropriate response to inputs. In general, the validation process can be considered to be part of a larger process of system improvement. So, our philosophy is that validation should not only provide a statement of validity, but also serve as the mechanism for finding the invalid parts of the system and how to repair them.

The best possible means to predict the validity of a system is to subject it to actual conditions for an arbitrary period of time. It is hoped that during this time the system would be subjected to all potential situations and its performance could be measured from the correctness of its response to these. However, this begs two questions: 1) How can it be assured that the system will in fact see all potential sets of inputs to which it may reasonably have to respond and 2) how can it be easily and definitively determined that the responses are correct? Our work attempts to answer these questions and here we present an approach to validating intelligent systems effectively as well as efficiently.

The heart of the presented methodology is a TURING test–like systematic interrogation of the system being validated.

Buchanan and Shortliffe [6] describe a Turing test approach in their evaluation of MYCIN that shares some commonalities with our technique. While comprehensive in nature, they do not attempt to generalize it to serve for all knowledge-based systems. Our approach formalizes the technique as much as feasible and the result is a generic, albeit conceptual, one to be usable by many types of knowledge-based systems. The main differences between Buchanan's Turing test and the one suggested here are as follows. 1) They use two separate panels of different experts, respectively for the test case solving session and for the session that rates the "goodness" of the solutions. Our approach uses only one. 2) They do not formally consider the fact that the expert's competences may vary within the different experts as well as within the different test cases; our approach does.

Our procedure results in a near-complete automation of the validation process. Complete automation, in our opinion, will remain an elusive goal because of the need to employ expert validators.

A. Steps in the Proposed Validation Process

The process of intelligent system validation can be said to be composed of the following related steps [15].

1) *Test case generation:* Generate and optimize a set of test input combinations (test data) that will simulate the inputs to be seen by the system in actual operation. We refer to the pairs [*TestData, ExpectedOutput*] as test cases. There are two competing requirements in this step: 1) *coverage* of all combinations of inputs that are possible, thus expanding the number of test cases to ensure completeness in coverage and 2) *efficiency* minimizing the number of test cases to make the process practical. A workable compromise between these constraints is central to our proposed technique.
2) *Test case experimentation:* Since intelligent systems emulate human expertise, it is clear that human opinion needs to be considered when evaluating the correctness of the system's response. But human experts can vary in their competence, their own self-image and their bias for or against automation. Thus, it is important that an efficient method exist to fairly evaluate the correctness of the system's outputs given imperfect human expertise. This step, therefore, consists of exercising the resulting set of test data (from step 1) by the intelligent system as well as by the one or more validating experts in order to obtain and document the responses to each test data by the various sources.
3) *Evaluation:* This step interprets the results of the experimentation step and determines errors attributed to the system and reports it in an informal way.
4) *Validity assessment:* This step analyzes the results reported above and reaches conclusions about the validity of the system.
5) *System refinement:* In order to improve the final system, this step provides guidance on how to correct the errors detected in the system as a result of the previous four steps. This, hopefully, leads to an improved system.

These steps are iterative in nature, where the process can be conducted again after the improvements have been made. Fig. 1 illustrates the steps outlined.

The methodology to implement these steps and its application to a frequently used kind of rule-based systems will be described in the following sections. A detailed description of all steps as well as the research behind this work can be found in [16].

II. GENERATION OF TEST CASES

One standard that does exist, however impractical it may be in most cases, is the exhaustive testing of the knowledge-based system. That is, generate a set of test cases which covers all contingencies possible in the operation of the system. For systems which have more than a few inputs, the combinations of values of these inputs can be prohibitively large, thus making exhaustive testing quite impractical [10]. Nevertheless, it is not necessary in most cases to have a truly exhaustive set of test cases and yet still be able to test the system in a *functionally exhaustive* fashion. As stated by Chandrasekaran [7], the test cases should reflect the problems to be seen by the system.

A *functionally exhaustive* set of test cases can be made considerably smaller than a *naively exhaustive* set by eliminating functionally equivalent input values and combinations of input values which subsume other values. Nevertheless, even this functionally exhaustive set is usually too large for practical purposes. Thus, there is a need for further reduction. Of course, one has to pay for it with a loss of functional exhaustivity.

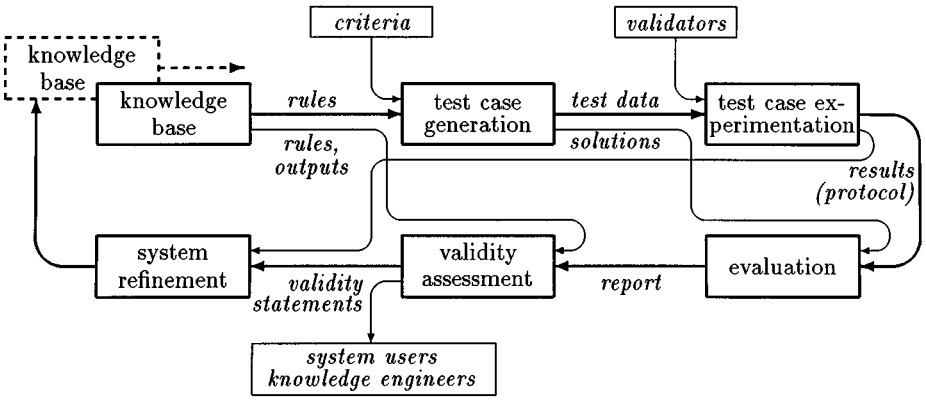

Fig. 1. Steps in the proposed validation process.

A reasonable way to reduce the functional exhaustive set of test cases is to use *validation criteria*, which can be domain-, input-, output-, expert-, validator-, or user-related in nature. These criteria are useful in determining a *test sufficiency level* for each test case of the functional exhaustive set. This test sufficiency level can be used as an indicator for the decision whether or not a given test case is really needed from a practical standpoint.

Due mainly to simplification reasons, but also because of its practical relevance, we consider rule-based systems with an input I of an m-dimensional "input space," in which each dimension is "atomic," i.e., not compound in any way and an output O of a set of possible output values.

The main test case generation idea described here is 1) to generate a "quasi exhaustive" set of test cases ($QuEST$) [3], [12] and 2) to define some validation criteria and to use them for a reduction of $QuEST$ down to a "reasonable" set of test cases ($ReST$) as described in [1].

A. Generation of Potential Test Cases

A frequently used kind of rule-based system, at least for classification problems, uses HORN clauses of the kind $conclusion \leftarrow \bigwedge_{i=1}^{n} attribute_i = value_i^j.\ conclusion$ is usually a single expression of the propositional calculus and the set of attributes forms the input space of the system.

Data Description: Formally, the rules can be described as follows.

- $S = \{s_1, s_2, \ldots, s_m\}$ is a set of variables s_i designating the *input sensor data* and ranging between $s_i^{\min}$ and $s_i^{\max}$ with a "normal value" s_i^{norm}.
- $E \subset \{[t_1, rel, t_2] : t_1 \in S, t_2 \in (S \cup \Re), rel \in \{<, \leq, =, \neq, \geq, >\}\}$[1] is a set of single *expressions* about sensor data.
- $F = \{f_1, f_2, \ldots, f_n\}$ is a set of outputs (*final conclusions*).
- $Int = \{int_1, int_2, \ldots, int_k\}$ is a set of *intermediate conclusions*.
- $R \subseteq \{[left, right] : right \in (E \cup I)^+, left \in F \cup I\}$[2] is a set of *rules* in which
  1) right-hand-side *right* designates the *if*-part of a rule;
  2) left-hand-side *left* designates the *then*-part of a rule.

[1] $\Re$ is the set of real numbers.

[2] M^+ expresses $M \cup (M \times M) \cup (M \times M \times M) \cdots = \bigcup_{i=1}^{\infty} M^i$

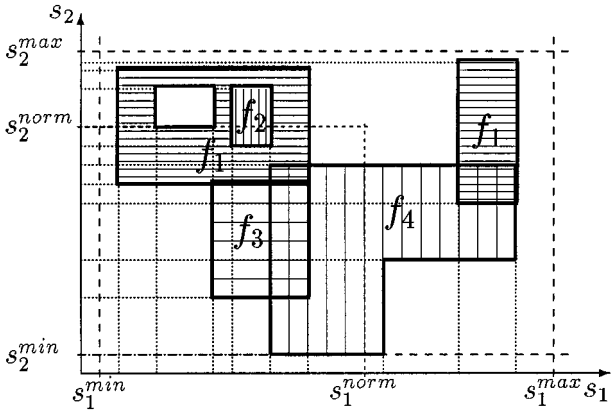

Fig. 2. Regions of influence for a two-input problem.

The input I of the system is formed by a set of points of the m-dimensional input space: $I = \{[s_1, \ldots, s_1, \ldots, s_m] : s_i^{\min} \leq s_i \leq s_i^{\max}\}$. The output O of the system is the set of final conclusions, i.e., $O = F$. A test case is a pair $[t_j, sol_j]$ with the test data $t_j = [s_1^j, s_2^j, \ldots, s_m^j] \in I$ and an associated solution $sol_j \in F$.

A *region of influence* is one (or several) convex subspace(s) of I formed by the intersection of the projection of the values of the sensors which have a direct effect on a particular final conclusion (f_i). Thus, values of the related sensors, which as a group fall within this region, will be able to identify a particular final conclusion. Fig. 2 shows the various regions of influence for a two input problem. In fact, these regions are expressed by the rules. Since an expert who expresses a rule does not necesarily care for other rules, it may happen that these regions overlap each other and/or that there are several different areas of the input space that are mapped to the same final conclusion.

The lines demarcating the regions from other regions are called the *region boundaries*.

The "quasi-exhaustive" set of test cases $QuEST$ has to meet the following requirements.

1) For each $f_i \in F$ there is at least one testcase in $QuEST$.
2) The test data t_j should be able to reflect the boundary conditions between different regions of influence.
3) The cardinality of $QuEST$ should be as small as possible. $QuEST$ does not consider the overlaps.

The fact that in case of overlaps one test case may serve for several final conclusions is utilized in the concept of the reasonable set of test cases $ReST$ (see next section).

The Approach: The process is based upon the following ideas:

1) break down the range of an input into non–overlapping subranges where its values are considered to be equivalent in terms of its effects on the final conclusions;
2) compute an initial set of potential test data P based upon combinations of values within these subranges;
3) sort these data into several sets P_i of data for each final conclusion f_i;
4) filter each P_i seperately by eliminating those test cases within each P_i that are subsumed by others.

This approach can be realized by the following steps.

Step 1: Computation of Dependency Sets: The first step is to compute the *rule dependency sets* $R_i \subseteq R$ and the *sensor dependency sets* $S_i \subseteq S$ for each $f_i \in F$. R_i only contains rules $r_k \in R$ and S_i only contains variables $s_k \in S$ on which f_i depends. This is easily carried out by tracing the rules backward from the final conclusions to the sensor inputs.

Step 2: Computation of Critical- and Δ-Values: A particular value of variable s_k is called *critical*, iff that value marks a difference in the effect of s_k on one of its dependent final conclusions. Critical values are determined by inspection of the rule left-hand sides, where values of s_k are described in relation to either constants or other variables. If an s_k is related to another variable s_j, then all the critical values of s_j must be considered also. All critical values of one variable s_k form the set of critical values S_k^{crit}. These critical values bound the subranges of functionally equivalent values for each variable and they define the regions of influence. For example, for a rule $f_3 \leftarrow (s_1 \leq 4) \wedge (s_1 > 2.5)$, the values 2.5 and 4 are elements of S_1^{crit}.

Next, we compute a Δ-value Δs_k for each $s_k \in S$. These values will allow us to surround the boundaries of the resulting regions of influence. Having all critical value sets S_k^{crit} we establish a Δ-value Δs_k for each s_k generated from S_k^{crit}. This is because a small change in a sensor input may distinguish two different final conclusions. One intuitive approach for computing Δs_k of each variable s_k is to ensure that Δs_k is half of the smallest difference between any "pair of critical values" that are members of S_k^{crit}. In case, for example $S_k^{crit} = \{2.5, 2.9, 4, 7.5\}$, the smallest difference is 0.4 and therefore $\Delta s_k = 0.2$. If $S_k^{crit} = \emptyset$, then Δs_k is set to the half of the smallest difference between any pairs of $\{s_k^{\min}, s_k^{norm}, s_k^{\max}\}$.

Step 3: Computation of the Sets of Potential Test Case Values (PTC-Values): The next step is to compute all sets V_{ij} $(0 < i \leq n, 0 < j \leq m)$ of all *PTC-Values* of s_j which contribute in any way to f_i. This has to be done by searching through each S_i and R_i. V_{ij} contains all *PTC-Values* of $s_j \in S$, which are in any way responsible for f_i. We have to look for each $s_j \in S$, whether s_j is an element of S_i or not and, if $s_j \in S_i$, whether there is a relationship between s_j and a fixed value or another s_k. There are three cases that have to be distinguished.

1) $s_j \notin S_i$
   In that case, s_j does not contribute to f_i and V_{ij} would be empty. Because each variable of a test data has to be assigned with a value, the *PTC-Value* of s_j is set to its normal value: $V_{ij} = \{s_j^{norm}\}$.
2) $s_j \in S_i$ and s_j is compared with at least one fixed value $\langle value \rangle$
   In that case there are three *PTC-Values* for each value s_j is compared with $s_j = \langle value \rangle - \Delta s_j$, $s_j = \langle value \rangle$ and $s_j = \langle value \rangle + \Delta s_j$. These three values have to be added to V_{ij}.[3]
3) $s_j \in S_i$ and s_j is compared with another variable s_k
   There are three possible subcases here.
   a) If $V_{ik} \neq \emptyset$, then define three *PTC-Values* for each $s_k \in V_{ik} \cap S_k^{crit}$, namely $s_j = s_k - \Delta s_k$, $s_j = s_k$ and $s_j = s_k + \Delta s_k$, which have to be added to V_{ij}.[4]
   b) If $V_{ik} = \emptyset$ and there is a joint interval of s_j and s_k, namely between $s^{int\_\min}$ and $s^{int\_\max}$, then create a *temporary set* TS with (note that $\Delta s_k = \Delta s_j$)

$$\begin{aligned} TS = \Big\{ & s_j^{\min}, s_j^{\min} + \Delta s_k, s_j^{norm} - \Delta s_k, s_j^{norm}, s_j^{norm} \\ & + \Delta s_k, s_j^{\max} - \Delta s_k, s_j^{\max}, s_k^{min}, s_k^{\min} \\ & + \Delta s_k, s_k^{norm} - \Delta s_k, s_k^{norm}, s_k^{norm} \\ & + \Delta s_k, s_k^{\max} - \Delta s_k, s_k^{\max} \Big\}. \end{aligned}$$

   Those elements of TS that are between $s^{int\_\min} - \Delta s_k$ and $s^{int\_\max} + \Delta s_k$ (inclusively) are added to both V_{ij} and V_{ik}.
   c) If $V_{ik} = \emptyset$ and there is no joint interval of s_j and s_k,[5] then both sets V_{ij} and V_{ik} will be equal to the set of their normal values: $V_{ij} = \{s_j^{norm}\}$ and $V_{ik} = \{s_k^{norm}\}$.

Step 4: The Set of All Potential Test Data: Having completed the procedure to calculate *PTC-Values* for all f_i leads us to n^*m sets V_{ij}.

P will be the union of all test data $t \in M(f_i)$ that can be created from each $M(f_i)$ by computing the cross product of the sets $V_{i1}, \ldots, V_{im}$

$$P = \bigcup_{i=1}^{n} M_i = \bigcup_{i=1}^{n} (V_{i1} \times V_{i2} \times \cdots \times V_{im}) = \bigcup_{i=1}^{n} \prod_{j=1}^{m} V_{ij}.$$

Step 5: Minimizing the Set of All Potential Test Data: The huge cardinality of P[6] forces us to minimize the number of test cases and to find the minimized set of functionally necessary

[3]If there are values being created which go beyond the borders of s_j or s_k, then those values have to be substituted with the appropriate minimum or maximum value.

[4]Here, we do the same as above with values beyond the borders.

[5]In this case for each relation $(<, \leq, =, \neq, \geq, >)$ and any value of s_j and s_k we are able to decide whether the expression being treated is true or false. By the way, such cases indicate a mistake of knowledge acquisition, because such expressions are always true or always false (not depending on the values of the two sensor data variables). The expression can be removed in case of always being true, resp. the whole rule can be removed in case of always being false.

[6]Note that in the worst case, i.e., if all V_{ij} has the same cardinality c_i, the cardinality of only one M_i will be equal to the mth power of c_i! Nevertheless, if $card(S_i) \ll card(S)$ then the number of potential test data $(card(P_i))$ decreases significantly.

TABLE I
HANDLING OF SENSOR DATA VARIABLES BEING IN DIFFERENT RELATIONS TO VALUES OR ANOTHER SENSOR DATA VARIABLE

#	case expression	If the conclusion of the rule $r \in R_i$, which contains the considered expression should be **true** to make f_i happen: Take that tuple with	should be **false** to make f_i happen: Take that tuple with
1	$s_j < \langle val \rangle$	the largest value of s_j that is lower than $\langle val \rangle$	the smallest value of s_j that is greater than $\langle val \rangle$ and that tuple with a value of s_j that is equal to $\langle val \rangle$
2	$s_j \leq \langle val \rangle$	the largest value of s_j that is lower than $\langle val \rangle$ and that tuple with a value of s_j that is equal to $\langle val \rangle$	the smallest value of s_j that is greater than $\langle val \rangle$
3	$s_j = \langle val \rangle$	a value of s_j that is equal to $\langle val \rangle$	the smallest value of s_j that is larger than $\langle val \rangle$ and the tuple with the largest value of s_j that is lower than $\langle val \rangle$
4	$s_j \neq \langle val \rangle$	the smallest value of s_j that is larger than $\langle val \rangle$ and the tuple with the largest value of s_j that is lower than $\langle val \rangle$	a value of s_j that is equal to $\langle val \rangle$
5	$s_j \geq \langle val \rangle$	the smallest value of s_j that is greater than $\langle val \rangle$ and that tuple with a value of s_j that is equal to $\langle val \rangle$	the largest value of s_j that is lower than $\langle val \rangle$
6	$s_j > \langle val \rangle$	the smallest value of s_j that is greater than $\langle val \rangle$	the largest value of s_j that is lower than $\langle val \rangle$ and that tuple with a value of s_j that is equal to $\langle val \rangle$
7	$s_j < s_k$	the largest value of s_j that is lower than the joint value of s_k	the smallest value of s_j that is greater than the joint value of s_k and that tuple with a value of s_j that is equal to the joint value of s_k
8	$s_j \leq s_k$	the largest value of s_j that is lower than the joint value of s_k and that tuple with a value of s_j that is equal to the joint value of s_k	the smallest value of s_j that is greater than the joint value of s_k
9	$s_j = s_k$	a value of s_j that is equal to the joint value of s_k	the largest value of s_j that is lower than the joint value of s_k and that tuple with the lowest value of s_j that is greater than the joint value of s_k
10	$s_j \neq s_k$	the largest value of s_j that is lower than the joint value of s_k and that tuple with the lowest value of s_j that is greater than the joint value of s_k	a value of s_j that is equal to the joint value of s_k
11	$s_j \geq s_k$	the smallest value of s_j that is greater than the joint value of s_k and that tuple with a value of s_j that is equal to the joint value of s_k	the largest value of s_j that is lower than the joint value of s_k
12	$s_j > s_k$	the smallest value of s_j that is greater than the joint value of s_k	the largest value of s_j that is lower than the joint value of s_k and that tuple with a value of s_j that is equal to the joint value of s_k

ones ($QuEST$). As described in more detail below, to reduce the cardinality of P we first sort P into $n+1$ different subsets P_i $(0 \leq i \leq n)$ and then we minimize all the subsets P_i $(0 < i \leq n)$ separately, through which sets $P_i^\star$ will be generated. $QuEST$ is the union of each $P_i^\star$. P_0 represents the set of negative test data for all final conclusions, i.e., those test data that will not identify any final conclusion.

Step 5.1: Sorting P Into $n+1$ Subsets P_i: First we have to sort all m–tuples $t \in P$. Due to the necessity of keeping the two parts of the test case (test data t_j and solution f_j) together, we represent P as an $n+1$-tuple $[P_0, P_1, \ldots, P_n]$, where each P_i $(i > 0)$ contains those test data $t \in P$ that are positive ones for f_i and P_0 contains all test data $t \in P$ being negative for all f_i. In case a test case is positive for multiple f_i, it belongs to each of the associated P_i. We have to check for each $t \in P$,[7] whether the system maps to a final conclusion f_i. If any f_i is true, then t will be an element of P_i of positive test data for the proof of f_i, i.e., $[t, f_i]$ is a positive test case, otherwise t will be an element of the set of negative test data P_0.

Step 5.2: Minimizing All the Sets P_i: The approach for minimizing each set P_i individually (with the exception of P_0) consists of the following steps.

1) Segregate the largest possible subset P_{seg} of m–tuples which differ in only one value (let's say, s_j) from the set P_i. Thus, P_{seg} is a subset of the considered P_i that contains m–tuples with identical values at $m-1$ positions and different values at the remaining position.

[7]In former publications we checked only, whether a test data $t_j \in M_i$ maps to the associated conclusion f_i. As a result of the evaluation of our technology (see Section VI) we changed this strategy.

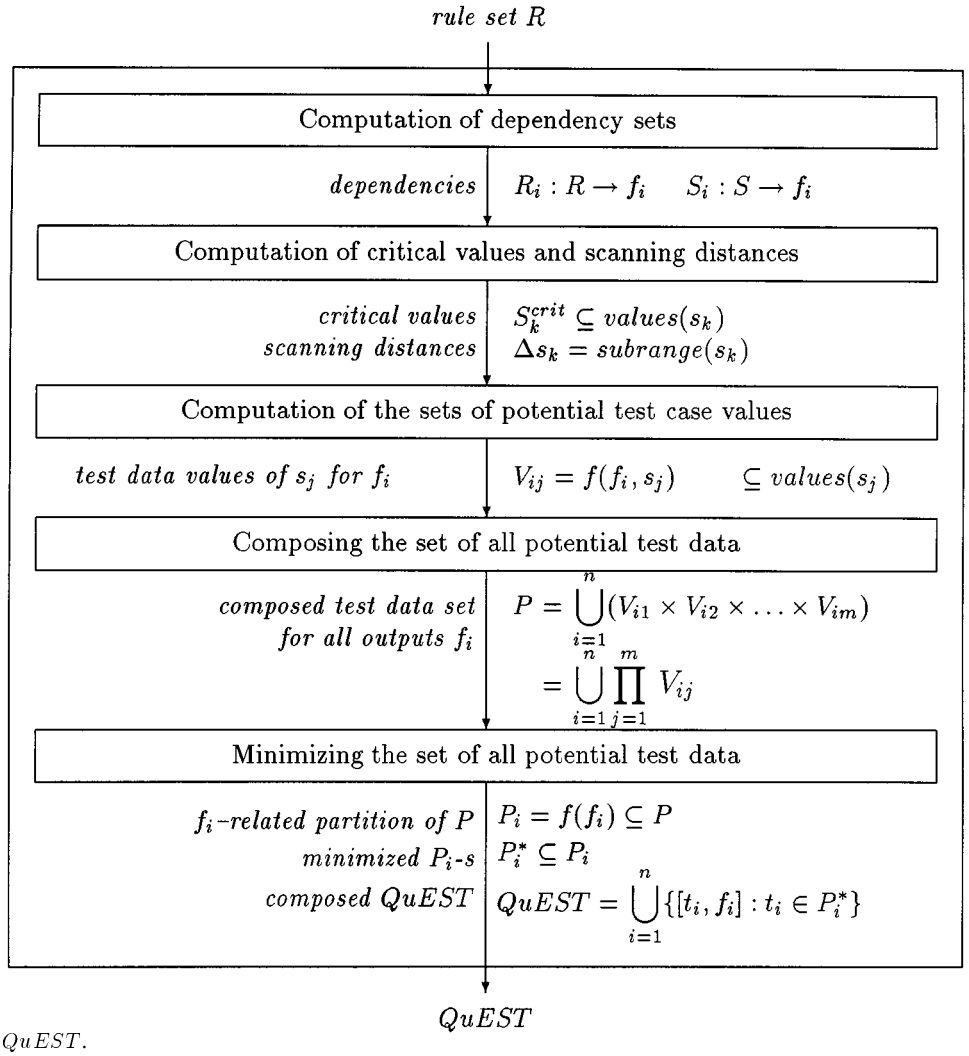

Fig. 3. Generation of $QuEST$.

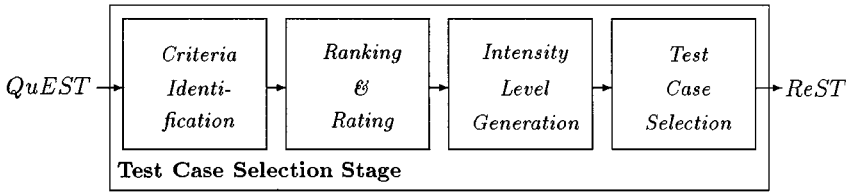

Fig. 4. Criteria-based test case selection for XPS validation.

2) With this segregated subset P_{seg}, look through R_i and gather all expressions in which s_j is compared to any value or another variable s_k. For each member of this subset, the relations between s_j and either s_k or a constant value must be one of the 12 cases shown on Table I. The result is that all potential test data that do not match any of the conditions described in Table I are removed from that subset. Therefore, all potential test data that are subsumed by other ones will be removed. The remaining set is P_{good}.
3) Let the new set P_i be the minimized set: $P_{i\_new} := (P_{i\_old} \setminus P_{seg}) \cup P_{good}$.
4) Repeat this (go to step 1) with P_{i_new} until no subset P_{seg} is creatable from P_{i_new}.
5) The minimized subset $P_i^\star$ is the set P_{i_new} computed in step 3).

This procedure should be carried out for each of the sets P_i. In the end, the minimized set of test cases able to test the system quasi exhaustively is the union of all sets $P_i^\star$, i.e., $QuEST = \bigcup_{i=1}^{n} P_i^\star$. To sum up and illustrate the technology of generating $QuEST$; see Fig. 3.

B. Criteria-Based Strategy to Reduce a Set of Potential Test Cases

This subsection describes a methodology to reduce the quasiexhaustive set of test cases $QuEST$ down to a "reasonable" (in the sense of "manageable" by a validation technology) set of test cases $ReST$. The steps of developing $ReST$ out of $QuEST$ are illustrated in Fig. 4.

The basic concept is that certain criteria about the system or associated factors influence the importance of each test case in

the $QuEST$. After the relevant criteria have been identified, they are ranked and rated. The ranking describes the quantitative relations inbetween the particular criteria; the rating describes a criterion's influence on the investigated domain for the considered output. Based on the results of this process, a test intensity level can be computed that forms the basis for reducing the quasi–exhaustive set of test cases $QuEST$ down to a "reasonable" set of test cases $ReST$. By removing those test cases that are of less importance than a certain threshold, the suite of test cases can be reduced significantly.

1) Criteria Identification: Abel developed a catalog of criteria (cf. [1] and [2]), which should be considered in order to answer the question of how important a certain test case is for the system's validity. This catalog contains conceptual criteria and human criteria, which are motivated by the scenarios of developing, using and validating the system. Abel classifies the criteria into two main groups:

a) Conceptual Criteria:

Domain Related Criteria (*DRC*) (criticality, complexity, sensitivity, domain coverage, domain robustness, …), Input Related Criteria (*IRC*) (criticality, sensitivity, characteristics, …) and Output Related Criteria (*ORC*) (probability, criticality, sensitivity, costs, robustness, …)

b) Human Criteria:

Expert Related Criteria (*ERC*) (competence, credibility, availibility, …), Validator Related Criteria (*VRC*) (objectivity, competence, independence, neutrality, …) and User Related Criteria (*URC*) (acceptable level of performance, maintainability, effectiveness, usability, …)

Identifying the relationships between a given criterion and its influence on the Test Case Selection Stage is best done by the user community, in collaboration with the development team and the expert. The involved individuals must establish a common ground about defining the criteria and how it should be used in the validation of the system. We found this to be beyond the scope of our work and leave it for future research.

2) Ranking and Rating: After indentifying the relevant criteria out of this catalog, Abel *et al.* (cf. [1]) suggest a **Criteria Ranking** as a first step toward the criteria-based reduction of test cases. The criteria having a measurable influence on the domain have to be identified. For each criteria, a rank has to be established by using a Criteria Assessment Scale. A rank expresses a criterion's importance related to the other ones.

3) Intensity Level Generation: The second step is a **Domain Assessment**, which uses the ranked domain–related criteria (DRC) as well as the output-related criteria (ORC). The result of it is a **Global Test Necessity Level** of the entire system and a **Local Test Necessity Level** for each of the system's outputs. Here, all the assessible characteristics of the whole domain and the different conclusions (outputs) have to be rated using the ranked criteria. We propose the use of the same kind of scale as for ranking and rating and a two–level–assessment: 1) assess/rate the domain using the ranked *DRC* and determine a **Global Test Necessity Level** TNL and 2) assess/rate all hypotheses separately using the ranked ORC and determine a **Local Test Necessity Level** $\boldsymbol{tnl(f)}$ for each final conclusion (output) f.

4) Test Case Selection: To answer the question of which of the test cases have to be selected out of the quasi-exhaustive set of test cases ($QuEST$) to become a member of the "reasonable" set of test cases ($ReST$), ABEL *et al.* suggest that a **Test Sufficiency Level** should be determined for each of the members of the $QuEST$. This can be performed by considering 1) the input space of the system; 2) the dependency sets of the outputs and the critical values of each input dimension; and 3) the regions of influence. By comparing this test sufficiency level of each test data with the test necessity level of its solution (a system's output), it is determined whether or not a test case of the $QuEST$ should belong to the $ReST$. Loosely speaking, the more a test case 1) has relevant input data that is relevant for other test cases as well; 2) has relevant input data that contributes to outputs with high rated local test necessity level; 3) has relevant input data within an important interval of its range; and 4) is situated near a border of a region of influence, the more this test case has a claim to become a member of $ReST$. For more detailed information about the quantitative considerations of test case selection see [1].

5) Formal Description of the Steps Above: Let us define ($k_D, k_O, n, m, r \in \mathbb{N}$) 1) k_D assessible Domain Related Criteria drc_i; 2) k_O assessible Output Related Criteria orc_i; 3) *one* assessment scale having an odd quantity ($r_{\max}$) of ordered scale values r ($1 \leq r \leq r_{\max}$); 4) n outputs (final conclusions) f_i, the domain "outputs;" 5) m sensor variables s_i, $S = \{s_1, s_2, \ldots, s_m\}$ is the domain's *Sensor Set*, the domain "inputs;" 6) n *Sensor Dependency Sets* S_i, ($S_i \subseteq S$, S_i is the set of all sensors f_i depends on); 7) n minimized (quasi–exhaustive) sets $P_i^\star$, ($P_1^\star \cup P_2^\star \cup \ldots \cup P_n^\star = QuEST$) of *positive test cases* ($P_i^\star$ contains well–selected test cases ([3]) implying f_i); 8) m sets S_i^{crit} of *critical values*; and 9) n^*m sets $V_{ij}(f_i, s_j)$ of *potential test case values*. The sets $P_i^\star$, S_i, S_i^{crit} and V_{ij} were generated during the Test Case Generation Stage as shown in Section II-A.

Assessing the Domain: Each criterion orc (resp. drc) is given a rank $\hat{r}(orc)$ (resp. $\hat{r}(drc)$) using the criteria assessment scale $1 \ldots r_{\max}$. As a result of using the complete "expressivity" of the assessment scale, the highest of these ranks should be $r_{\max}$. Since these values are normalized with respect to their maximum value, it does not have to (but Should) be $r_{\max}$. Having done this, there is a k_D–tuple of *DRC*—rankings $R_D = [\hat{r}_1(drc_1), \hat{r}_2(drc_2), \ldots, \hat{r}_{k_D}(drc_{k_D})]$ and a k_O–tuple of orc—rankings $R_O = [\hat{r}_1(orc_1), \hat{r}_2(orc_2), \ldots, \hat{r}_{k_O}(orc_{k_O})]$.

Global Test Necessity Level TNL: All the ranked *DRC* will be given criteria–dependent ratings $r(drc_i)$. The difference between a rating and a ranking is that the ranking describes the proportions among the criteria, i.e., how important a criterion is compared to the other ones, whereas the rating describes a criterion's influence on the investigated domain, i.e., how important a criterion is with respect to the domain.

The weights of the domain criteria $w(drc_i)$ are $w(drc_i) = \hat{r}(drc_i)^* r(drc_i)$ with $0 < w \leq r_{\max}^2$. The Global Test Necessity Level TNL has to be normalized to the maximum weight of all *DRC*, i.e., $TNL = \max(\{w(drc_i) | 1 \leq i \leq k_D\})/r_{\max}^2$.

Local Test Necessity Levels $tnl(f_i)$: Having ranked all *ORC*, the next step is to provide criteria–dependent ratings to all outputs of the domain as well. A k_O–tuple of ratings $R_i(f_i) =$

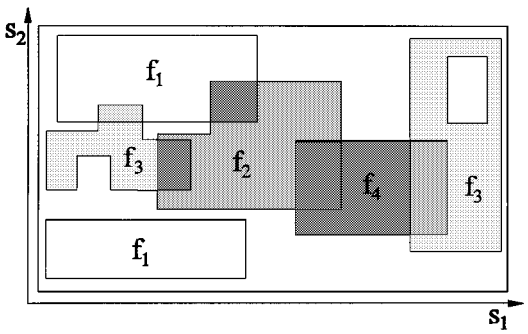

Fig. 5. Regions of influences with different test necessity levels.

$[r_{i1}, \ldots, r_{ik_O}]$ with $r_{ij} \in \{1, 2, \ldots, r_{\max}\}$ is generated for each output f_i.

These tuples can be represented as an $[n \times k_O]$–matrix

$$\mathbf{R} = \begin{pmatrix} R_1(f_1) \\ R_2(f_2) \\ \vdots \\ R_n(f_n) \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & \ldots & r_{1k_O} \\ r_{21} & r_{22} & \ldots & r_{2k_O} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n_1} & r_{n_2} & \ldots & r_{nk_O} \end{pmatrix}.$$

Each entry r_{ij} describes the rating of a criterion orc_j for the output f_i [8]. The ith ranked and summated row can be considered as an output's weight $w(f_i) = w(f_i) = \sum_{j=1}^{k_O} \hat{r}(orc_j)^* r_{ij}$, which is a description of its *Validation Necessity* as well. The (normalized) *Test Necessity Level* $tnl(f_i)$ of an output f_i is

$$tnl(f_i) = \frac{TNL^* w(f_i)}{\max(\{w(f_k) | 1 \le k \le n\})} = TNL^* \hat{w}(f_i).$$

Assessing the Test Cases : The results of the preceding "Ranking and Rating Sessions" are 1) *one* Global Test Necessity Level TNL $(0 < TNL \le 1)$ (2) n distinctive Local Test Necessity Levels $tnl(f_i)$ $(0 < tnl \le TNL)$. These have to be consulted in order to decide which test cases can be neglected in the following test case evaluation stage, i.e., which test cases are sufficient for system validation.

At first, a *Test Sufficiency Level* $tsl(t)$ for each test case $t \in QuEST$ has to be generated. Finally, $ReST$ will be fomed as $ReST = \{t | t \in P_i^\star \wedge tsl(t) \le tnl(f_i)\}$.

Computation of the Test Sufficiency Level for a Test Case t: The decision of which test cases are to remain in the $ReST$ is influenced by the domain's sensor dependency sets. Imagine a two input problem, the outputs can be represented as areas, *regions of influence* ([3]), that are situated in the domain's input space delimited by the sensors' value ranges (see Fig. 5).

The following obvious statements can intuitively be realized.

1) If a sensor belongs to more sensor dependency sets than another sensor, then this one is possibly of a higher importance for the validation process.
2) Sensors belonging to sensor dependency sets of higher rated outputs are certainly more important than other sensors.
3) There can be more or less important intervals of a sensor's value range.
4) The importance of a sensor value can be influenced by its distance to the domain's and/or sensor's boundaries.

The degrees of "greyness" of the different outputs $(f_1, \ldots, f_4)$ are descriptions of their importance $(tnl(f_i))$. The regions of influences are commonly partly covered by others. The darker distinctive areas in the input space are the more important is the corresponding interval of the sensor's value range.

$tsl(t)$ can be generated by 1) getting the *weights of all sensors* $\hat{w}(s)$ using all $tnl(f)$ and 2) getting the *weights of all sensor values* $\hat{w}(s^{val})$ originated during the Test Case Generation Stage using all $tnl(f)$ and generating a *test sufficiency level* $tsl(t)$ by using $\hat{w}(s)$ and $\hat{w}(s^{val})$ for each $t \in QuEST$.

Computing $\hat{w}(s_j)$ is done based on first computing $w(s_j) = \sum_{i=1}^{n} C_i^* tnl(f_i)$ with $c_i = 0$ iff $s_j \notin S_i$ and $c_i = 1$ otherwise. The extreme cases of the equation above are 1) no output depends on the value of the considered sensor variable s_j, i.e., $w(s_j) = 0$ and 2) all outputs depend on the value of the considered sensor variable s_j, i.e., $w(s_j) = \sum_{i=1}^{n} tnl(f_i)$. The normalized weight $\hat{w}(s_j)$ is, i.e., $\hat{w}(s_j) = w(s_j)/\max(\{w(s_k) | 1 \le K \le M\})$.

Computing $\hat{w}(s_j^{val})$ can be done as[9]

$$w(s_j^{val}) = \sum_{i=1}^{n} C_{ij}^* tnl(f_i)$$

$$\text{with } c_{ij} = \begin{cases} 0, & \text{iff } s_j \notin S_i \\ 1, & \text{iff } s_j \in S_i \wedge s_j^{val} \notin (V_{ij} \cup S_j^{crit}) \\ 2, & \text{iff } s_j \in S_i \wedge s_j^{val} \in (V_{ij} \setminus S_j^{crit}) \\ 3, & \text{iff } s_j \in S_i \wedge s_j^{val} \in (S_j^{crit} \setminus V_{ij}) \\ 4, & \text{iff } s_j \in S_i \wedge s_j^{val} \in (V_{ij} \cap S_j^{crit}). \end{cases}$$

Here, the test necessity levels of the outputs f_i are weighted depending on to what degree the considered sensor data s_j contributes to it.

1) In case it does not contribute at all, the weight c_{ij} is zero. Otherwise, i.e., in case it contributes, this degree depends on whether or not there are values of s_j in the (sub-)set of potential test case values V_{ij} and the set of critical values S_j^{crit}.
2) In case it has no particular value in either the potential test case value (sub-)set V_{ij} or the set of critical values S_j^{crit}, it is set to one.
3) In case it has such a value in V_{ij} but not one in S_j^{crit}, it is two.
4) In case it has such a value in S_j^{crit} but not one in V_{ij}, it is three.
5) In case it has a value in both, it is four.

The normalized weight $\hat{w}(s_j^{val})$ is related to the maximum weight of all sensor values of sensor s_j (let there be k different values of s_j), i.e., $\hat{w}(s_j^{val}) = w(s_j^{val})/\max(\{w(s_j^x) | 1 \le x \le k\})$.

[8] A summed row is an expression of the (local) importance of the concerned output referring to the other ones. A summed column is an expression of the (local) influence of the concerned criterion referring to the other ones. The higher the sums the higher their importances (resp. influences).

[9] The c_i and c_{ij} values (1 … 4) are very intuitively chosen here based on the authors' experience. If the investigated application field of the expert system allows other weights of c_i (resp. c_{ij}), the factors can be adjusted.

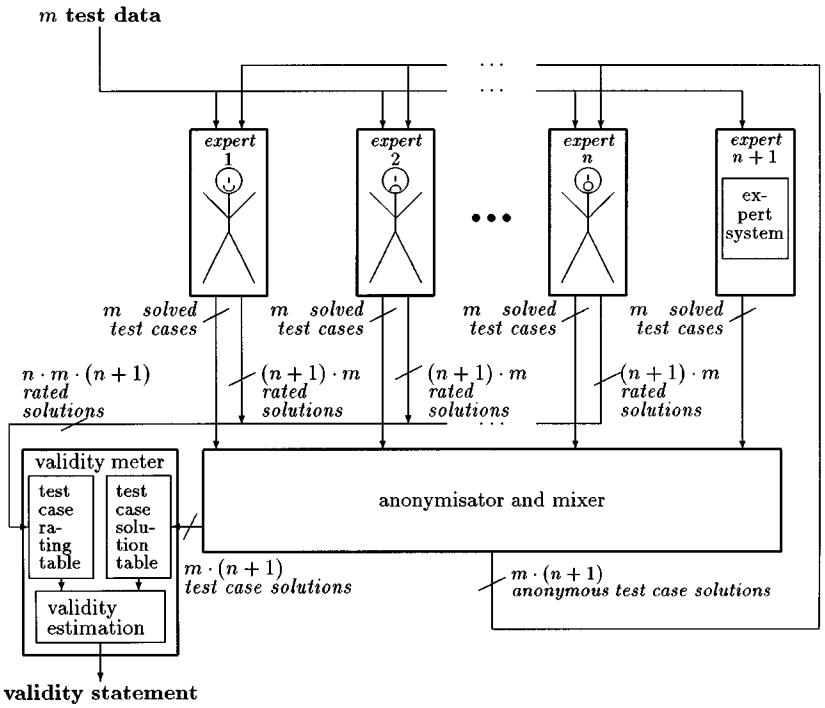

Fig. 6. Survey of the TURING test to estimate an AI system's validity.

Computing $tsl(t)$*:* A test data is considered as an m-tuple of sensor values [3], which can be represented as $t_k = [s_1^k, s_2^k, \ldots, s_m^k]$ The weight of a test data t_k is $w(t_k) = \sum_{j=1}^{m} \hat{w}(s_j)^* \hat{w}(s_j^k)$.

The *Test Sufficiency Level* $tsl(t_k)$ of a test case t_k is

$$tsl\,(t_k) = 1 - \frac{w\,(t_k)}{\max\left(\{w\,(t_x)\,|1 \le x \le ||P_i^\star||\}\right).}$$

Summary: **A Test Necessity Level** value $tnl(f)$ for each output f representing the necessity of a validation of output f is generated. In other words: How extensively do I have to investigate an output to get a credible validity statement?

A Test Sufficiency Level value $tsl(t)$ is generated for each test data of the test cases in $QuEST$. This value, $tsl(t)$, expresses the sufficiency of t for a validation of a final conclusion. In other words: Up to which level is a test case sufficient for a validation of an output having a certain Test Necessity Level?[10]

To get a sufficient, but nevertheless **credible validity statement** for an output f, it's only necessary to check f with test cases t having a $tsl(t)$ that is lower or equal to $tnl(f)$. In other words: During the Test Case Evaluation Stage we only need to take into consideration test cases having a lower or equal Test Sufficiency Level tsl than the Test Necessity Level tnl of a particular output, i.e., $ReST = \{t | t \in P_i^\star \wedge (tsl(t) \le tnl(f_i))\}$.

III. TURING TEST EXPERIMENTATION WITH TEST CASES

Here, we describe some ideas on developing a validity statement based on a TURING test—like methodology with a "reasonable" set of test cases $ReST$.

[10]Note, that the Test Suficiency Level is 1—"normalized weight", i.e., a low level means a high sufficiency.

A. Proposed Technique—An Overview

The suggested methodology is quite similar in concept to the TURING test. It involves the system to be validated, a panel of n experts and the set $ReST$ to produce 1) a test case associated validity statement for each test data t_j and 2) a global validity degree of the entire system.

The idea of the TURING test methodology, as illustrated in Fig. 6, is divided into four steps: 1) solving of the test cases by the panel as well as the system; 2) randomly mixing the test case solutions and removing their authorship; 3) rating all (anonymous) test case solutions; and 4) evaluating the ratings.

B. Solving Test Cases

Solving m test data sets t_j by n (human) experts $e_1, \ldots, e_n$ and the system e_{n+1} leads to $m^*(n+1)$ *solved test cases* $[t_j, e_i, sol_{ji}]$ with the solution sol_{ji}. sol_{ji} is either a "real" output or "unknown" by its provider: $sol_{ji} \in O \cup \{unknown\}$. The output set O is formed by *all* upcoming solutions: $O = \pi_{outp}(\mathcal{E}_{i+1}) \cup \pi_{outp}\left(\bigcup_{i=1}^{n} \mathcal{E}_i\right) = \pi_{outp}\left(\bigcup_{i=1}^{n+1} \mathcal{E}_i\right)$..

C. Making the Solutions Anonymously

To ensure that the human experts not be aware of a solution's author (and especially which is the system's solution and which is their own), each one of the n human experts gets the $m^*(n+1)$ upcoming *solved test cases* without any information about the authorship.

D. Rating the Solutions

With a rating $r \in \{0, 1\}$ and a certainty $c(r) \in \{0, 1\}$ the experts express their opinion about the solution ($r = 1$: "correct", $r = 0$: "incorrect") and their confidence to be valid ($c = 1$: "sure," $c = 0$: "unsure"). Additionally, they have the chance to express a lack of competence by $r = norating(c(norating) =$

0). Each rating r_{ijk} is assigned to a solution sol_{jk} of the expert $e_k(1 \le k \le n+1)$ of a test data $t_j(1 \le j \le m)$ and an evaluating (human) expert $e_i(1 \le i \le n)$ and has the certainty c_{ijk}, i.e., the two subscripts of the considered solution are preceeded by a subscript that indicates the rating expert.

E. Evaluating the Ratings

This procedure is done by the *validity meter*, which has $m^*(n+1)$ *solved test cases* and $n^*m^*(n+1)$ *rated test case solutions* (each solved test case is rated by n experts). The output of the evaluation procedure described here is a validity degree $0 \le v(t_j) \le 1$ for each test data $t_j \in \pi_{inp}(ReST)$. The procedure is performed by calculating an average rating of the system's solution by the experts, each one weighted by the considered expert's competence for t_j as well as by his/her certainty. Additionally, it provides a global validity of the entire system v_{sys}.

1) Estimating an Expert's Competence: The first step toward a validity statement is to estimate the competence of each expert. We prefer to do that for each expert and for each test case separately due to the fact that not all experts are equally competent for a given test case and a certain expert's competence is not equal for all test cases. The competence estimation of an expert e_i for a test data t_j is based on 1) his/her own evaluation to be competent; 2) his/her certainty while rating other experts' solutions; 3) his/her consistency in the solving and the rating process;[11] 4) his/her stability;[12] and 5) the other experts' ratings of his/her solution.

The competence estimation of an expert e_i for a test data t_j is based on …

1) …self-evaluation of competence, as indicated by giving the solution $unknown$ and/or the rating $norating$

$$slf\_est(e_i,t_j) = \frac{1}{2} ord(sol_{ji} \neq unknown) + \frac{1}{2}\frac{1}{n}\sum_{k=1,k\neq i}^{n+1} ord(r_{ijk} \neq norating);$$

2) …his/her certainty while rating other experts' solutions, as indicated by the ratio between the number of certain ratings and the number of ratings altogether: $crt\_est(e_i,t_j) = 1/n \sum_{k=1,k\neq i}^{n+1} c_{ijk}$;
3) …his/her consistency in the solving and the rating process, as indicated by the rating of the own solution: $cns\_est(e_i,t_j) = r_{iji}$;
4) …his/her stability, as indicated by the certainty of the own solution's rating: $stb\_est(e_i,t_j) = c_{iji}$;
5) …the other experts' ratings of his/her solution, as indicated by their average ratings, weighted by their certainties: $frgn\_est(e_i,t_j) = 1/\left(\sum_{k=1,k\neq i}^{n} c_{kji}\right) \sum_{k=1,k\neq i}^{n} (c_{kji}{}^* r_{kji})$.

There are three main sources of competence estimation: 1) intentional reflection: self-estimation and certainty (slf_est, crt_est); 2) nonintended reflection: consistency and stability (cns_est, stb_est); and 3) external (foreign) competence estimation ($frgn\_est$). These are taken into account equally as

$$cpt(e_i,t_j) = \frac{1}{3}{}^*\left(\frac{1}{2} slf\_est(e_i,t_j) + \frac{1}{2} crt\_est(e_i,t_j)\right) + \frac{1}{3}{}^*\left(\frac{1}{2} cns\_est(e_i,t_j) + \frac{1}{2} stb\_est(e_i,t_j)\right) + \frac{1}{3}{}^* frgn\_est(e_i,t_j).$$

2) Estimating the System's Validity: Thus, the average rating of the system's solution by the experts, each one weighted by the considered expert's competence for t_j as well as by his/her certainty is

$$v_{sys}(t_j) = \frac{1}{\sum_{i=1}^{n}\left(cpt(e_i,t_j)^* c_{ij(n+1)}\right)} {}^* \sum_{i=1}^{n}\left(cpt(e_i,t_j)^* c_{ij(n+1)}{}^* r_{ij(n+1)}\right).$$

This is an estimation of the system's validity for a test data t_j.

The entire expert system's validity v_{sys} can be estimated by the average local validity $v_{sys}(t_j)$ for each test case t_j: $v_{sys} = 1/m \sum_{j=1}^{m} v(t_j)$. Depending on some domain- and user-related validation criteria (see previous section) each system can be associated with a minimum validity $v^{\min}$, which is a threshold value for the validity statement. That is, of course, the objective of this part of the research: The system is called **valid**, iff $v \ge v^{\min}$ and **invalid** otherwise.

IV. Evaluation and Validity Assessment

Depending on its purpose there are several ways to express a system's validity. Besides the two validity assessment that are the result of the experimentation, there can be calculated at least two other useful validity assessments. The four ways to express validity that are considered useful by the authors are as follows.

1) **Global (average) validity**

$$v_{sys} = 1/(|ReST|) \sum_{j=1}^{(|ReST|)} v_{sys}(t_j).$$

2) **Validities associated with outputs**

$$v_{sys}(sol_k) = 1/(|T_k|) \sum_{[t_j,sol_k]\in T_k} v_{sys}(t_j)$$

$$(T_k = \{[t_j,sol_k] \in ReST : t_j \in \pi_{inp}(ReST), [t_j,sol_k] \in \mathcal{E}_{n+1}\}).$$

3) **Validities associated with rules**

$$v(r_l) = 1/|T_l| \sum_{[t_j,sol_k]\in T_l} v_{sys}(t_j)$$

$$(T_l = \{[t_j,sol_k] \in ReST : t_j \in \pi_{inp}(ReST), [t_j,sol_k] \in \mathcal{E}_{n+1}, t_j \; uses \; r_l\}).$$

4) **Validities associated with test data**

$$v_{sys}(t_j) = \frac{1}{\sum_{i=1}^{n}\left(cpt(e_i,t_j)\cdot c_{ij(n+1)}\right)} \cdot \sum_{i=1}^{n}\left(cpt(e_i,t_j)\cdot c_{ij(n+1)}\cdot r_{ij(n+1)}\right).$$

[11]Does he/she give his/her own solution good marks?

[12]Is he/she certain while rating his/her own solution?

1) and 2) might be useful for (potential) system users and/or managers, 3) for system developers, namely knowledge engineers, and 4) is the basis of formal system refinement.

V. System Refinement

The basic idea to refine the system consists of finding the "guilty" rules and systematically replacing them by "better ones." A rule is considered "guilty" if a conclusion that received "bad marks" by the expert panel forms the conclusion-part (*then*-part) of this rule. This is performed by the following steps.

1) Finding guilty rules: Analyze which test cases used which rules and which validity degree has been associated with the system's solution of these test cases. The "last" rule, i.e., the rule which has exactly this solution in its then-part, is called "guilty." Generate an "optimal solution" for each of these test cases, which is either the system's solution or a solution provided by a human expert.
2) Reduction of the set of guilty rules: Repair those guilty rules that have the same optimal solution for all test cases using this rule.
3) Replacing the if-part of the remaining guilty rules: Repair the remaining guilty rules by a reduction system that systematically constructs (one or more) new rule(s) as a substitute for the guilty rule.
4) Recompiling the new rules and removing unused rules: Recompile the upcoming substitutes by utilizing rules that infer intermediate hypotheses. Remove unused rules.

Each of these steps is explained in the following subsections.

A. Finding Guilty Rules

All rules having a conclusion part which is a final solution sol_k, are the subject of the following considerations.

1) There is a rule-associated validity for each of these rules $v(r_l) = 1/|T_l| \sum_{[t_j, sol_k] \in T_l} v_{sys}(t_j)$.
2) There is a set T_l^* of test cases with test data $t_j \in \pi_{inp}(T_l)$ and all solution parts which came up in the experimentation by any e_i $T_l^* = T_l \cup \{[t_j, sol(e_i, t_j)] : \exists [t_j, sol_k] \in T_l\}$.
3) T_l^* can be split into subsets $T_{l1}^*, \ldots, T_{lp}^*, \ldots, T_{lm}^*$ according to their different solution parts $sol_1, \ldots, sol_p, \ldots, sol_m$.
4) Analogously to $v_{sys}(sol_k)$, a validity $v(r_l, sol_p)(1 \le p \le m)$ of each solution $sol_1, \ldots, sol_p, \ldots, sol_m$ can be computed, but only based on the test cases of T_{lp}^*—can be computed

$$v(r_l, sol_p) = \frac{1}{\left|T_{lp}^*\right|} \sum_{[t_j, sol_p] \in T_{lp}^*} \frac{1}{\sum_{i=1}^{n} (cpt(e_i, t_j) \cdot c_{ijq})} \cdot \sum_{i=1}^{n} (cpt(e_i, t_j) \cdot c_{ijq} \cdot r_{ijq}).$$

5) The "optimal validity" of r_l is the maximum of all $v(r_l, sol_p)$ among the solutions sol_p occurring in T_l^*. The associated solution is the "optimal solution" sol_{opt} of r_l: $v_{opt}(r_l, sol_{opt}) = \max(\{v(r_l, sol_1), \ldots, v(r_l, sol_m)\})$. $v_{opt}(r_l)$ is an upper limit of the reachable rule-associated validity of r_l. If $v_{opt}(r_l, sol_{opt}) > v(r_l)$, there is a solution within T_l^* which got better marks by the experts than the system's solution. Thus, if $v_{opt}(r_l, sol_{opt}) > v(r_l)$, r_l is a guilty rule.

B. Reduction of the Set of Guilty Rules

If all test cases in T_l that used a guilty rule r_l have the same optimal solution sol_k that was different from the system's solution, the conclusion-part of this rule has to be substituted by sol_k.

$\forall [t_j, sol_{k_1}] \in T_l : sol_s$ is "optimal solution" to $t_j \Rightarrow r_l :$ (if-part $\rightarrow sol_k$) $\hookrightarrow$ (if-part $\rightarrow sol_s$).

C. Replacing the If-Part of the Remaining Guilty Rules

1) T_l of a guilty rule r_l is split into subsets $T_l^1, \ldots, T_l^s, \ldots, T_l^n$ according to the solution sol_s for each t_j that got the highest validity $v(r_l, sol_s)$. The new if-part(s) of the new rule(s) instead of a remaining guilty rule r_l are expressions $e_i \in E$ of a set of p new alternative rules $\{r_l^1, r_l^2, \ldots, r_l^p\}$ for each T_l^s and will be noted as a set of sets $P_l^s = \{\{e_1^1, \ldots, e_{p_1}^1\}, \{e_1^2, \ldots, e_{p_2}^2\}, \ldots \{e_1^p, \ldots, e_{p_p}^p\}\}$. The corresponding rule set of P_l^s is $r_l^1 : \bigwedge_{i=1}^{p_1} e_i^1 \rightarrow sol_s$ $r_l^2 : \bigwedge_{i=1}^{p_2} e_i^2 \rightarrow sol_s$ $\ldots r_l^p : \bigwedge_{i=1}^{p_p} e_i^p \rightarrow sol_s$.
2) Pos is the set of Positions (dimensions of the input space), at which the $t_j \in \pi_{inp}(T_l^s)$ are **not** identical. The generation of the if-parts P_l^s is managed by a *Reduction System*, which is applied to Triples $[T_l^s, Pos, P_l^s]$ until Pos becomes the empty set $\emptyset$.
3) The starting point of the reduction is $[T_l^s, Pos, P_l^s]$ with $P_l^s = \{\{(s_1 = s_1^{ident}), \ldots, (s_q = s_q^{ident})\}\}$. $s_1, \ldots, s_q$ are those positions, where all test data $t_j \in \pi_{inp}(T_l^s)$ have the same (identical) value s_i^{ident} and Pos is the set of the remaining positions: $Pos = \{s_i : \neg\exists (s_i = s_i^{ident}) \in P_l^s\}$.

Table II shows the reduction rules used to reconstruct the remaining guilty rules. The reduction system terminates if the situation $[T_l^s, \emptyset, P_l^s]$ is reached. A deeper discussion on the reduction technique can be found in [16].

D. Recompiling the New Rules and Removing the Unused Rules

In case the *if*-part of a new rule contains a subset of expressions that is the *if*-part of another rule having an intermediate solution as the *then*-part, this subset has to be replaced by the corresponding intermediate solution

$$\begin{aligned} &\exists r_i : (if\text{-part}_1 \rightarrow int_1) \\ &\exists r_j : (if\text{-part}_1 \wedge if\text{-part}_2 \rightarrow int\text{-}or\text{-}sol) \Rightarrow \\ &\quad r_j : (if\text{-part}_1 \wedge if\text{-part}_2 \rightarrow int\text{-}or\text{-}sol) \\ &\quad \hookrightarrow (int_1 \wedge if\text{-part}_2 \rightarrow int\text{-}or\text{-}sol). \end{aligned}$$

TABLE II
REDUCTION RULES TO CONSTRUCT BETTER RULES SYSTEMATICALLY

Reduction rules

R1 • $pos \in Pos$, s_{pos} has a value set with no well-defined $\leq$ relation

• $\{s^1_{pos}, \ldots, s^m_{pos}\}$ are the values of s_{pos} occurring in T^s_l $\Rightarrow$

$$[T^s_l, Pos, \{p_1, \ldots, p_n\}] \hookrightarrow$$

1. $[T^{s,1}_l \setminus \{[t_j, sol_s] \in T^s_l : s_{pos} \neq s^1_{pos}\}, Pos \setminus \{pos\}, \bigcup_{i=1}^{n} p_i \cup \{(s_{pos} = s^1_{pos})\}]$

2. $[T^{s,2}_l \setminus \{[t_j, sol_s] \in T^s_l : s_{pos} \neq s^2_{pos}\}, Pos \setminus \{pos\}, \bigcup_{i=1}^{n} p_i \cup \{(s_{pos} = s^2_{pos})\}]$

• • •

m. $[T^{s,m}_l \setminus \{[t_j, sol_s] \in T^s_l : s_{pos} \neq s^m_{pos}\}, Pos \setminus \{pos\}, \bigcup_{i=1}^{n} p_i \cup \{(s_{pos} = s^m_{pos})\}]$

Continue with each $T^{s,i}_l$ $(1 \leq i \leq m)$ separately.

R2 • $pos \in Pos$, s_{pos} has a value set with a well-defined $\leq$-relation

• s^{min}_{pos} is the smallest value of s_{pos} within T^s_l

• s^{max}_{pos} is the largest value of s_{pos} within T^s_l $\Rightarrow$

$$[T^s_l, Pos, \{p_1, \ldots, p_n\}] \hookrightarrow$$

$$[T^s_l, Pos \setminus \{pos\}, \bigcup_{i=1}^{n} p_i \cup \{(s_{pos} \geq s^{min}_{pos}), (s_{pos} \leq s^{max}_{pos})\} \cup S_{excl}]$$

S_{excl} is the set of excluded values for s_{pos}, which have to be mapped to a solution different from sol_s because of belonging to some other T^v_u with $v \neq s$:

$$S_{excl} = \{(s_{pos} \neq s^j_{pos}) : \exists [t_j, sol_s] \in T^s_l \ \exists [t_m, sol_v] \in T^v_u (v \neq s) \text{ with } \forall p \neq pos \ ((s^j_p = s^m_p) \text{ and } (s^{min}_{pos} < s^m_{pos} < s^{max}_{pos}))\}$$

Lastly, we remove those rules, which have an intermediate hypothesis as the *then*-part, which is not used in the *if*-part of any rule

$$\exists r_i : (if\text{-part}_1 \rightarrow int_1) \neg$$
$$\exists r_j : (int_1 \wedge if\text{-part}_2 \rightarrow int\text{-}or\text{-}sol) \Rightarrow$$
$$r_i : (if\text{-part}_1 \rightarrow int_1) \hookrightarrow \emptyset.$$

VI. EVALUATION AND ANALYSIS OF THE METHODOLOGY

We applied part of the above technique to a nontrivial expert systems to determine its effectiveness as well as its usefulness. More specifically, this exercise put into practice the concept of the $QuEST$. We empirically evaluated the validity of our hypothesis that the $QuEST$ represents an equivalent, yet much smaller, set of test cases to that of the exhaustive set (EST). We studied how well the $QuEST$ was able to identify seeded errors in the knowledge base when test cases in the $QuEST$ were executed by the system under test and its response was judged by experts.

For reasons of practicality, the small but robust expert system chosen dealt with classification of bird types. Called the Ornithologist, it represents a classification expert system. The system is rule based and consists of 71 rules capable of classifying 65 different types of birds, i.e., there are six rules that infer intermediate results. The Ornithologist uses up to 14 different inputs, but it does not need all of them at all times. Nevertheless, at least two inputs are always necessary to identify a bird. A typical rule looks like this:

IF $size$ > 17 AND $size$ < 24 AND $winter\_b$ AND $bill\_g$ AND ab_duck
THEN $spezies$ = American Black Duck, Anas rubripes

Here, $winter\_b$ is an intermediate hypothesis and both $bill\_g$ as well as ab_duck are boolean inputs.

The cardinality of the exhaustive set of test cases was theoretically computed to be 35,108,736,000—a clearly unmanageable number. This was computed as the combination of the system's 14 inputs and their possible values. Most of the inputs were discrete, each having two or three possible values, except for the length of the bird. For this input, the continuous range was decomposed into intervals of 0.10 inch. Alternatively, the $QuEST$ generated by our proposed technique resulted in 317 test cases, a large but manageable number. Moreover, the number of test cases in the set P_0 was determined to be an additional 1063.

Thirty-six errors were seeded in the expert system. In lieu of having an expert, the original system was deemed to be valid by definition and thus served as the "expert". A seeded error was said to be properly identified if the original (unchanged) system provided a different answer from the modified system when a test case that made use of the purposely modified rule was presented to both systems. This discrepancy would cause the knowledge engineer to investigate the rule which the test cases tested and presumably find the error.

Of the 36 errors seeded, 26 were properly found. Ten errors were not detected for various reasons. Nine of these ten would have been detected if minor adjustments to the $QuEST$ generating procedure had been made. Thus, these were considered to be easily correctable errors in the procedure. The last undetected error, however, brought to light a serious limitation that could only be corrected by including the set of test cases P_0 as part of the $QuEST$. Of course, this had the effect of increasing the size of the $QuEST$ from 317 to 1380 cases. While significantly larger than before, it still represents a manageable number, especially when the test for reasonableness is yet to be done which will further reduce that number. Thus, the conclusion was that with the inclusion of P_0, we had high confidence that the $QuEST$ indeed represented the equivalent of the exhaustive set of test cases. For detailed information on the performed experiments, refer to [19]; a more general analysis is presented in [16].

A. Analysis of Technique

The problem of classification is a generalization of the diagnosis problem, where the symptoms can be classified as those observed when a specific type of malfunction occurs in the system being monitored and/or diagnosed. In fact, some of the ideas contained in this technique originally came about from work by one of the authors in a diagnostic expert system for large turbine generators [11].

In this section we extrapolate the results obtained and discuss the amount of effort and cost involved in implementing this procedure in a full expert system development project. Again, the estimates are based on the experiences of one of the authors in the above-mentioned project.

This analysis focuses on the following issues: 1) applicability of this technique to real-world expert systems; 2) effort involved in applying this technique to the validation of a reasonably-sized expert system; and 3) computational cost (complexity) of the technique.

1) Applicability to Real World Expert Systems: We believe that this technique could be used for most rule-based systems employed in the real world. We define real-world expert systems as those either in use, or contemplated for use in solving real problems. One main group of such systems and the one on which we focus, represents those that deal with engineering or other technical problem/opportunities. Examples of such are diagnosis, classification, monitoring, control, design, analysis, data filtering and others. These are systems for which rule-based expert systems represent a viable problem-solving technique.

But there are some caveats. 1) The expert system must be rule-based. 2) Since the test case generation technique is based on the rule-base structure, it is necessary that the internal rule structure of the expert system be made visible to the test personnel. 3) It is furthermore assumed that the system has already been verified through acceptable means to ensure its consistency, completeness and satisfaction of specifications. 4) The specification must spell out the validation criteria in detail as part of the specification. Lacking this, the $ReST$ may not be significantly smaller than the $QuEST$, making the process much more costly.

2) Cost of Implementing the Described Technique: Of course, cost is an important consideration when testing any type of system. In this section we analyze the probable costs, making estimates based on the bird classification example described above and our personal experiences in validating real world diagnostic expert systems. We do not include the cost of developing the tools that implement the techniques described.

We are assuming that the $QuEST$ of 1380 test cases can be further reduced by 75% to a Reasonable Set of Test cases ($ReST$). This set would now be reduced to 345 test cases. While still a nontrivial number, we feel that this number is quite conservative and based on the experience cited above, we believe that it could even be reduced to 10% or less of the $QuEST$.

The costs involved would be broken down into the following. 1) The cost of each expert in responding to the test cases. 2) The cost for a knowledge engineer to exercise the system with the 345 cases. 3) The cost of each expert analyzing the cases/responses. 4) The cost of someone managing the test cases and setting up the web site to facilitate testing. 5) The cost of a test engineer compiling the results. These will be analyzed individually in the following.

The following quantitative considerations are based on the authors' personal experience of how long an expert needs to solve a test case and review a given solution. For particular applications, these estimations may have to be modified to fit the application domain.

For item 1, we estimate that each expert, if truly an expert, will take no more than 5 min to do each test case. That implies 28.75 person-hours. Including breaks, distractions and all other obstacles to concentration, we shall assume a person-week of effort for each expert. Naturally, some test cases may take longer than 5 min, but we think most of them will be answered almost immediately, thus making this a conservative number.

Item 2: The time required for a knowledge engineer to exercise the system should be minimal if the test cases are automatically generated as suggested by our methodology. Continuing our conservative trend, we shall assume one person-day of effort.

For item 3, assuming a panel of three experts, there would be 1380 test cases and responses to analyze by each of the experts. This number is obtained by multiplying the number of test cases (345) by the number of responses provided for each one (by the three experts plus the expert system). We assume that the experts are working independently and there is no discussion among them. The cases/responses could be accessed via the web and the responses could also easily be handled in the same manner. Given that the experts already have seen the test cases and should be familiar with them, we assume 1 min for each response. This equates to 23 h per expert. Once again, we shall assume that interruptions, breaks etc. will result in a total of one person-week for each expert.

Item 4 is fairly minimal, as this is quite mundane work done by a technician. We assume one person-day to maintain our conservative philosophy.

Lastly, item 5, compilation of results would add another person-day, assuming the availability of computerized tools to analyze and manipulate the results. This task would also include the knowledge engineer inspecting questionable results.

In summary, the effort spent by the panel of experts is 6 person-weeks and that of the knowledge engineer/technician is three person-days. We assume a costing rate of $200 000 per annum for the experts and $150 000 per annum for knowledge engineer and technician. Assuming 250 working days per year (50 weeks $\times$ 5 days), it equates to $800 per day for each expert ($4000 per week) and $600 per day for KE/technician ($3000 per week).

The total, then, is estimated to be $25 800.

While the Ornithologist is not a large system, it is not a trivial one. There are many systems in the real world that approximate its size and complexity. Furthermore, the figures are quite conservative, from the time expended to the number of test cases ultimately used.

3) Computational Complexity of System: A problem that can become a limiting factor for the adoption of the proposed methodology is its computational complexity. The most complex of the components is that which generates the $QuEST$ and the $ReST$. It turns out that the generation of $QuEST$ is a very complex issue, but this procedure is performed by machines, i.e., without any human support. This it the price for minimal work left to humans. We feel it is worth it to pay a high machine complexity cost in order to have minimal work left to humans.

Since the generaton of $ReST$ has been especially introduced to limit the number of test cases for the experimentation, its complexity is a function of the specified criteria and their rating and ranking. This can be difficult to predict. However, it can be as simple as linear if the criteria are independent from each other, or as complex as quadratic if the criteria are interdependent. These are considered tractable. The complexity of $QuEST$, on the other hand, might be a real problem at first sight. It depends exponentially on the number of inputs. The exponent is determined by the type of the input. 1) Adding a new input that has just two different values doubles the number of test cases. 2) adding a new input that has n different (discrete) values multiplies the cardinality of $QuEST$ by n. 3) adding an input with a continuous range of values, the cardinality of $QuEST$ will be muliplied by the number of its critical values (see Section II).

On first view, there seems to be a way out of this dilemma by dividing large knowledge bases into parts that are independent from each other with respect to the inputs. Thus, we can compute the $QuEST$s of these "sub-knowledge bases" separately. By inspecting our generation procedure for $QuEST$, we found out that this cannot lead to a limitation of the complexity, because this procedure already considers these dependencies und uses them to minimize $QuEST$.

Nevertheless, even NP-complete problems can be solved in a reasonable time if the N is limited. Thus, segmentation (modularization) of the knowledge base can be used, not to reduce the complexity of the algorithm, but rather, to reduce the N of the problem. Such a technique depends on a chunk of knowledge being independent of other chunks. This technique was also used successfully by Gonzalez *et al.* ([11]) to do a manual generation of test cases.

Again, since $QuEST$ can get very large we introduced the concept of $ReST$, which can be limited to any requested maximum number of test cases. Of course, the reliability of the upcoming validity statements heavily depends on the coverage of the domain with test cases, i.e., also on their number. Thus, one has to find a reasonable compromise between a minimal number of test cases and a most reliable resulting validity statement.

On one hand, generating test cases is the procedure with the highest computational complexity, but on the other hand, it is not the most expensive part of the methodology.

For performing the test case experimentation, it doesn't matter how long it takes to calculate $QuEST$-this is a procedure that is performed automatically, i.e., without any human support. It just needs computer ressources and we admit, this can become a limitation. However, to hire human experts is typically much more expensive than employing computer resources. And the objective of the test case generation procedure is to derive a test case set that is as small as possible to limit the costs of the human resources.

REFERENCES

[1] T. Abel and A. J. Gonzalez, "Utilizing criteria to reduce a set of test cases for expert system validation," in *Proc. 10th Florida AI Res. Symp. (FLAIRS-97)*, Dankel, Ed., Daytona Beach, FL, May 1997, pp. 402–406.

[2] ——, "Influences of criteria on the validation of AI systems," in *Proc. 5. Leipziger Informatik-Tage 1997 (LIT-97)*, Jantke and Grieser, Eds., Sept. 1997, pp. 43–48.

[3] T. Abel, R. Knauf, and A. J. Gonzalez, "Generation of a minimal set of test cases that is functionally equivalent to an exhaustive set, for use in knowledge-based system validation," in *Proc. 9th Florida AI Res. Symp. (FLAIRS-96)*, Stewman, Ed., Key West, FL, May 1996, pp. 280–284.

[4] A. Auzins, J. Barzdins, J. Bicevskis, K. Cerans, and A. Kalninf, "Automatic construction of test sets: Theoretical approach," in *Baltic Computer Science*, Barzdins and Bjorner, Eds. Berlin, Germany: Springer-Verlag, 1991, vol. 502, Lecture Notes in AI, pp. 286–359.

[5] B. W. Boehm, "Verifying and validating software requirements and design specifications," *IEEE Trans. Softw.*, vol. 1, pp. 75–88, 1984.

[6] B. G. Buchanan and E. H. Shortliffe, *Rule-Based Expert Systems—The MYCIN Experiments of the Stanford Heuristic Programming Project*. Reading, MA: Addison-Wesley, 1985.

[7] B. Chandrasekaran, "On evaluating AI systems for medical diagnosis," *AI Mag.*, vol. 4, no. 2, pp. 34–37, 1983.

[8] F. Coenen and T. Bench-Capon, *Maintainance of Knowledge-Based Systems*. New York: Academic, 1993.

[9] F. Coenen, B. Eaglestone, and M. Ridley, *Validation, Verification and Integrity in Knowledge and Dada Base Systems: Future Directions*, 1999, vol. 25, pp. 297–311.
[10] A. J. Gonzalez and D. D. Dankel, *The Engineering of Knowledge-based Systems—Theory and Practice*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[11] A. J. Gonzalez, U. Gupta, and R. B. Cianese, "Performance evaluation od a large diagnostic expert system using a heuristic test case generator," *Eng. Applicat. Artif. Intell.*, vol. 9, no. 3, pp. 275–284, 1996.
[12] J. Herrmann, K. P. Jantke, and R. Knauf, "Using structural knowledge for system validation," in *Proc. 10th Florida AI Res. Symp. (FLAIRS-97)*, Dankel, Ed., Daytona Beach, FL, May 1997, pp. 82–86.
[13] K. P. Jantke, R. Knauf, and T. Abel, "The Turing test approach to validaton," in *Proc. Workshop Validation, Verification & Refinement of AI Systems and Subsystems (W32), Int. Joint Conference on Artificial Intelligence (IJCAI-97)*, Terano, Ed., Nagoya, Japan, Aug. 1997, pp. 35–45.
[14] K. P. Jantke, R. Knauf, and A. Stephan, "Validation von Anwendungssoftware: Von der Forschung zum Marktfaktor," in *Proc. of 5. Leipziger Informatik-Tage 1997 (LIT-97)*, Jantke and Grieser, Eds., Sept. 1997, pp. 1–21.
[15] R. Knauf, T. Abel, K. P. Jantke, and A. J. Gonzalez, "A framework for validation of knowledge based systems," in *Proc. Int. Workshop Aspects Intelligent Systems Validation*, Grieser, Beick, and Jantke, Eds., Ilmenau, Germany, January 1998.
[16] R. Knauf, "Validating Rule-Based Systems—A Complete Methodology," Shaker, Aachen, Germany, ISBN 3-8265-8293-4, 2000.
[17] S. Lee and R. M. O'Keefe, "Developing a strategy for expert systems verification and validation," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 643–655, Apr. 1994.
[18] P. Meseguer, "A new method to checking rule bases for inconsistencies: A Petri net approach," in *Proc. Eur. Conf. AI (ECAI'90)*, Aiello, Ed., Stockholm, Sweden, Aug. 1990, pp. 437–442.
[19] J.-E. Michels, "Validating a validation tool—experiences with a test case providing strategy," Practicum Dept. Elect. Comput. Eng., Univ. Central Florida, Orlando, 1998.
[20] P. Meseguer and E. Plaza, "The VALID project: Goals, development and results," *Int. J. Intell. Syst.*, vol. 9, no. 9, pp. 867–892, 1994.
[21] R. M. O'Keefe and S. Lee, "Developing a strategy for expert system verification and validation," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 643–655, Apr. 1994.
[22] R. M. O'Keefe and D. E. O'Leary, "Expert system verification and validation: A survey and tutorial," *Artif. Intell. Rev.*, vol. 7, pp. 3–42, 1993.
[23] P. Stager, "Validation in complex systems: Behavioral issues," in *Verification and Validation of Complex Systems: Human Factors Issues*, ser. F: Computer and System Sciences. Berlin, Germany: Springer, 1993, vol. 110, NATO ASI, pp. 99–114.
[24] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. LIX(236), pp. 433–460, 1950.
[25] A. Vermesan and F. Coenen, *Validation and Verification of Knowledge Based Systems. Theory, Tools and Practice*, A. Vermesan and F. Coenen, Eds. Norwell, MA: Kluwer, 1999.
[26] J. A. Wise and M. A. Wise, "Basic considerations in verification and validation," in *Verification and Validation of Complex Systems: Human Factors Issues*. ser. F: Computer and System Sciences, Wise, Hopkin, and Stager, Eds. Berlin, Germany: Springer-Verlag, 1993, vol. 110, NATO ASI, pp. 87–95.
[27] N. P. Zlatareva and A. Preece, "State of the art in automated validation of knowledge-based systems," *Expert Syst. Applicat.*, vol. 7, no. 2, pp. 151–168, 1994.

Rainer Knauf was born in Erfurt, Germany, in 1963. He received the Diploma degree in electrical engineering in 1987, the Ph.D. degree in computer science in 1990, and the Habilitation degree in 2000 from the Ilmenau Technical University, Ilmenau, Germany.

From 1987 to 1990, he was an Assistant for Research and Teaching in artificial intelligence at the Ilmenau Institute of Technology. Since 1990, he has held several positions there and presently holds an Associate Professorship (Privatdozent) as the Chair of Artificial Intelligence with the Faculty of Computer Science and Automation. His research is focused on the evaluation of intelligent systems.

Avelino J. Gonzalez received the B.S. and M.S. degrees in electrical Engineering from the University of Miami, Coral Gables, FL, in 1973 and 1974, respectively, and the Ph.D. degree in electrical engineering from the University of Pittsburgh, Pittsburgh, PA, in 1979.

He worked in various engineering, administrative, and management capacities with the Westinghouse Electric Corporation from 1974 to 1986. While at Westinghouse, he worked extensively in modeling and simulation programs for power generation systems. His most significant accomplishment was the development of the Generator Artificial Intelligence Diagnostics System (GenAID) in 1985. Variations of his pioneering work are still in use today. In 1986, he accepted an apointment as faculty member in the Computer Engineering Department (now the School of Electrical Engineering and Computer Science), University of Central Florida, Orlando. His area of specialization is intelligent systems, and modeling of human behavior in simulations. He has authored several articles on the subject, and has written two textbooks.

Thomas Abel was born in Gräfenhainichen, Germany, in 1964.

He is a Senior Consultant with GFT Technologies AG, Ilmenau, Germany. His fields of interest include information technology, software engineering, knowledge management, quality assurance, and quality management.